MFE-IT

Référence : of-DEV-MERN26

Formation Full Stack React Node.js

Développeur Full Stack JavaScript : concevoir une application web moderne avec React et Node.js — du front-end au back-end (pile MERN)

Durée : 6 jours | Heures : 42 h | Tarif : 2 900 € HT

En présentiel ou à distance · Groupe de 1 à 3 · Session garantie dès 1 inscrit · ~70 % de pratique

DESCRIPTION

Cette formation conduit un développeur maîtrisant déjà les bases du web à concevoir des applications complètes avec la pile JavaScript moderne, de l'interface React jusqu'au serveur Node.js. Elle part du principe que le participant sait programmer : l'enjeu n'est pas d'apprendre un langage, mais d'assembler un écosystème — composants React, gestion d'état, API Express ou GraphQL, base de données, authentification, tests et déploiement — en une application cohérente et maintenable. Le fil conducteur reste toujours la pratique : au fil de six journées à forte dominante atelier (environ 70 % de travaux pratiques sur un projet fil rouge MERN), le participant construit, teste, sécurise puis met en production une application réelle, en s'appuyant sur les outils et les versions en usage (React 19, Vite, Node.js, Express, Apollo, Prisma…).

OBJECTIFS PÉDAGOGIQUES

À l'issue de cette formation, le participant sera capable de :

- Concevoir des interfaces React modernes et évolutives à l'aide des hooks et du rendu par composants.
- Gérer l'état d'une application et la récupération de données asynchrones (TanStack Query, Redux Toolkit, Zustand).
- Construire une API REST ou GraphQL avec Node.js et Express, structurée et sécurisée.
- Modéliser et interroger des données via un ORM, aussi bien en base NoSQL (MongoDB) que relationnelle (PostgreSQL).
- Mettre en place une authentification robuste (JWT, OAuth 2.0) et se prémunir des vulnérabilités web courantes.
- Sécuriser ses évolutions par une stratégie de tests unitaires, d'intégration et de bout en bout.
- Automatiser le déploiement continu et superviser une application en production.
- Structurer un projet full stack selon des principes d'architecture éprouvés.

PRÉREQUIS

- Bonne maîtrise de HTML et de CSS.
- Pratique confirmée de JavaScript, idéalement dans sa version moderne (ES6 et suivantes).
- Solides bases de programmation orientée objet.
- Aucune expérience préalable de React, de Node.js ou d'un ORM n'est requise.

PUBLIC VISÉ

- Développeur front-end ou back-end souhaitant couvrir l'ensemble de la chaîne full stack JavaScript.
- Développeur venant d'une autre pile (PHP, Java, .NET…) désireux de monter en compétence sur l'écosystème React / Node.js.
- Intégrateur ou développeur web voulant passer du site vitrine à des applications dynamiques connectées à une API.
- Profil technique amené à concevoir, reprendre ou faire évoluer une application MERN au sein d'une équipe.

PROGRAMME DÉTAILLÉ

La formation est à forte dominante atelier (environ 70 % de pratique) et s'appuie sur un projet fil rouge MERN — une application complète construite, enrichie et déployée progressivement au fil des six journées.

Module 1 — Panorama du développement full stack et mise en place de l'environnement

Le point de départ : comprendre l'architecture d'une application web moderne avant d'écrire la moindre ligne. On situe le rôle du développeur full stack, on clarifie le dialogue client-serveur et la logique de la pile MERN (MongoDB, Express, React, Node.js), puis on compare les deux grands styles d'API — REST et GraphQL — pour savoir quand privilégier l'un ou l'autre. La journée installe ensuite un environnement de travail complet et outillé (Node.js, npm, Visual Studio Code, Vite) et présente les outils qui accélèrent le cycle de développement moderne (SWC, esbuild, nodemon).

Atelier pratique : *initialiser un projet full stack de zéro, configurer Vite et le rechargement automatique du serveur, et valider que le front-end et le back-end communiquent.*

Module 2 — Développement front-end avec React.js

Le cœur de la partie visible de l'application. On pose les fondations de React (JSX, DOM virtuel, flux de données unidirectionnel), puis on exploite les apports des versions récentes (React 19 : composants serveur, Actions et hook use(), rendu concurrent et Suspense, avec le React Compiler). On structure un projet avec Vite, on met en forme l'interface (CSS Modules, Tailwind CSS) et surtout on maîtrise les hooks, véritable colonne vertébrale du React moderne : useState, useEffect, useReducer, useMemo, useCallback et la mémoïsation des composants.

Atelier pratique : *construire une interface composée de plusieurs composants réutilisables, gérer leur état local, puis traquer et supprimer les rendus superflus à l'aide des hooks de mémoïsation.*

Module 3 — Gestion de l'état et récupération des données

Dès qu'une application grandit, la circulation de l'information devient l'enjeu central. On distingue l'état serveur de l'état client et on adopte les bons outils pour chacun : TanStack Query pour la donnée asynchrone (cache, revalidation, états de chargement), Redux Toolkit et son middleware pour l'état global structuré, Zustand comme alternative légère. On relie enfin l'interface au serveur en consommant des API REST et GraphQL (fetch, Axios, Apollo Client).

Atelier pratique : *brancher l'interface sur l'API du projet, mettre en cache et rafraîchir les données avec TanStack Query, puis comparer l'approche avec un store Redux Toolkit.*

Module 4 — Développement back-end avec Node.js

Passage de l'autre côté : le serveur. On explique le modèle événementiel et non bloquant qui fait la singularité de Node.js, puis on construit une API REST avec Express.js. On met en place la chaîne de middlewares (authentification, validation, journalisation), on durcit l'API (Helmet, limitation de débit) et on ouvre sur GraphQL en créant une API avec Apollo Server.

Atelier pratique : *créer une API REST complète avec Express, y ajouter des middlewares de validation et de journalisation, puis exposer la même ressource via une requête GraphQL.*

Module 5 — Bases de données et ORM

Toute application a besoin de mémoire. On confronte l'approche documentaire de MongoDB aux bases relationnelles (PostgreSQL, MySQL) et on apprend à choisir selon le besoin. On modélise les données dans les deux paradigmes, on intègre Mongoose côté NoSQL et Prisma côté SQL, puis on aborde ce qui sépare une base lente d'une base rapide : optimisation des requêtes et stratégies d'indexation.

Atelier pratique : *modéliser le schéma du projet, réaliser les opérations de lecture et d'écriture via Mongoose puis via Prisma, et mesurer l'effet d'un index sur une requête.*

Module 6 — Authentification et sécurité

Une application connectée doit savoir qui fait quoi, et se défendre. On met en place une authentification par jeton JWT, on délègue l'identité via OAuth 2.0 (connexion Google, GitHub), on protège les mots de passe (bcrypt) et on valide systématiquement les données entrantes. On passe ensuite en revue les vulnérabilités web les plus fréquentes — injection XSS, injection SQL — et leur prévention, sans oublier la configuration CORS.

Atelier pratique : *sécuriser l'API du projet avec un flux d'authentification JWT, ajouter une connexion OAuth, puis corriger deux failles volontairement introduites (XSS et injection).*

Module 7 — Tests et qualité logicielle

Livrer vite suppose de pouvoir modifier sans crainte. On construit une stratégie de tests à plusieurs niveaux : tests unitaires et d'intégration avec Vitest, tests de composants avec React Testing Library, tests de bout en bout avec Cypress. On mesure la couverture, on éprouve la solidité des tests par la mutation (Stryker) et on simule les appels réseau avec MSW.

Atelier pratique : *écrire un jeu de tests couvrant un composant React et un endpoint d'API, puis automatiser un parcours utilisateur complet avec Cypress.*

Module 8 — Déploiement et intégration continue (CI/CD)

Une application n'existe vraiment qu'une fois en production. On optimise d'abord la construction pour la mise en ligne (Vite, esbuild), on passe en revue les plateformes d'hébergement (Netlify, Vercel, Heroku, Render, AWS), puis on automatise tout le cycle avec un pipeline GitHub Actions. Enfin, on met en place la supervision applicative (Sentry, Datadog) pour détecter et diagnostiquer les incidents.

Atelier pratique : *construire une version optimisée du projet, la déployer via un pipeline GitHub Actions, puis brancher une remontée d'erreurs en production.*

Module 9 — Temps réel et passage à l'échelle

Certaines applications doivent réagir instantanément et encaisser la charge. On implémente la communication en temps réel (WebSockets, Socket.IO), on introduit la mise en cache et les files d'attente avec Redis, on aborde la répartition de charge et le clustering, puis la gestion des tâches asynchrones et différées (Bull, Agenda).

Atelier pratique : *ajouter une fonctionnalité temps réel au projet (notifications ou messagerie instantanée), puis déporter un traitement lourd dans une file d'attente Redis.*

Module 10 — Architecture et bonnes pratiques

Le module de synthèse : faire tenir l'ensemble dans le temps. On structure un projet full stack en séparant clairement les responsabilités, on met en œuvre des patrons éprouvés (Repository, injection de dépendances), on s'ouvre à la conception orientée domaine (DDD), on externalise la configuration (dotenv) et on documente l'API pour l'équipe et les intégrateurs (Swagger, Postman).

Atelier pratique : *réorganiser le projet fil rouge selon une architecture en couches, documenter son API avec Swagger et rédiger une collection Postman de référence.*

POINTS FORTS DE LA FORMATION

- Une pile MERN complète et à jour (React 19, Node.js, Express, GraphQL, Prisma) couverte de bout en bout.
- Environ 70 % de pratique sur un projet fil rouge : du composant d'interface jusqu'à la mise en production.
- Un spectre large — front-end, back-end, données, sécurité, tests, déploiement — pour devenir réellement autonome en full stack.
- Petit groupe de 1 à 3 participants, session garantie dès 1 inscrit.

MÉTHODES PÉDAGOGIQUES

Format et déroulement

La formation est dispensée en présentiel ou à distance via une classe virtuelle interactive, avec un contenu adaptable aux besoins de votre projet. La répartition théorie/pratique est d'environ 30 %/70 %.

Format ultra-personnalisé MFE-IT

Chaque session accueille de 1 à 3 participants, garantissant un accompagnement hautement individualisé. Un entretien préalable permet d'adapter le contenu au profil de chaque participant. Les sessions inter-entreprises sont garanties dès 1 inscrit (aucun risque de report sauf cas de force majeure).

Évaluation des compétences

Tout au long de la formation, le formateur évalue la progression à travers des mises en situation et des travaux pratiques. À l'issue, une attestation de réussite est remise à chaque participant.

Suivi post-formation

Pendant un mois après la formation, chaque participant peut contacter les formateurs MFE-IT pour toute question relative à la mise en œuvre des acquis. Une réponse est apportée sous 48 heures ouvrées.

Accessibilité

MFE-IT s'engage à accueillir les personnes en situation de handicap. Contact : contact@mfe-it.com.

INFORMATIONS PRATIQUES

Ressources du formateur

- Démonstrations structurées alignées sur le programme détaillé
- Énoncés d'exercices et corrigés tout au long de la formation
- Un projet fil rouge MERN et un environnement technique prêts à l'emploi
- Validation des acquis par le formateur à la fin de chaque atelier

Certification et validation

À l'issue de la formation, une attestation est envoyée par e-mail précisant les objectifs, la nature, la durée et les résultats de l'évaluation. Un certificat de réalisation peut être fourni sur demande.

Financement

Cette formation est financée directement par l'entreprise (non éligible au CPF). Un devis personnalisé est établi selon le nombre de participants et le format retenu.